Configuring UARTs

Tuesday, March 31, 2020 1:40 PM

Objectives:	- Learn how to configure the UARTs for point to point communication - Learn about data encoding

Configuring the PIC32 UART

- Baud rate
- Number of data bits
- Parity, if used
- Number of stop bits
- Handshake protocol, if used
- Interrupt Modes, if used
- Error Handling, if used

UART SFRs

UxMODE - Mode Register

- Enables the UART module
- Enables the IrDA module
- Enables the Wake, ABAUD, and Loop-back functions
- Enables the UxRTS and UxCTS pins
- Configures the UxRTS pin for the desired mode
- Configures the polarity of the UxRx pin
- Selects the type of baud rate - high speed or standard
- Selects the number of data bits, parity, and stop bits

UxSTA - Status & Control Register

- Selects the transmission interrupt mode
- Selects the receive interrupt mode
- Enables or Disables the UART transmission
- Controls the Address Detect mode
- Indicates status conditions
    - Tx or Rx buffer state
    - Parity errors
    - Framing & overflow errors

UxTXREG - Data to be transmitted

UxRXREG - Data received

UxBRG - Baud rate register - stores the value corresponding to the desired baud rate

UxMODE

bit 15 **ON**: UARTx Enable bit(1)
bit 13 **SIDL**: Stop in Idle Mode bit
bit 12 **IREN**: IrDA Encoder and Decoder Enable bit
bit 11 **RTSMD**: Mode Selection for UxRTS Pin bit
bit 9-8 **UEN:** UART Function Enable bits
bit 7 **WAKE**: Enable Wake-up on Start bit Detect During Sleep Mode bit
bit 6 **LPBACK**: UARTx Loopback Mode Select bit
bit 5 **ABAUD**: Auto-Baud Enable bit
bit 4 **RXINV**: Receive Polarity Inversion bit
bit 3 **BRGH**: High Baud Rate Enable bit
bit 2-1 **PDSEL**: Parity and Data Selection bits
bit 0 **STSEL**: Stop Selection bit - 1 or 2 stop bits

UxSTA

bit 24 **ADM_EN**: Automatic Address Detect Mode Enable bit
bit 23-16 **ADDR**: Automatic Address Mask bits
bit 15-14 **UTXISEL**: TX Interrupt Mode Selection bits
    11 = Reserved, do not use
    10 = Interrupt is generated and asserted while the transmit buffer is empty
    01 = Interrupt is generated and asserted when all characters have been transmitted
    00 = Interrupt is generated and asserted while the transmit buffer contains at least one empty space
bit 13 **UTXINV**: Transmit Polarity Inversion bit
bit 12 **URXEN**: Receiver Enable bit
bit 11 **UTXBRK**: Transmit Break bit (Start bit followed by twelve '0' bits, followed by Stop bit
bit 10 **UTXEN**: Transmit Enable bit
bit 9 **UTXBF**: Transmit Buffer Full Status bit (read-only)
    1 = Transmit buffer is full
    0 = Transmit buffer is not full, at least one more character can be written
bit 8 **TRMT**: Transmit Shift Register is Empty bit (read-only)
bit 7-6 **URXISEL**: Receive Interrupt Mode Selection bit
    11 = Reserved
    10 = Interrupt flag bit is asserted while receive buffer is 3/4 or more full (has 6 or more data characters)
    01 = Interrupt flag bit is asserted while receive buffer is 1/2 or more full (has 4 or more data characters)
    00 = Interrupt flag bit is asserted while receive buffer is not empty (has at least 1 data character)
bit 5 **ADDEN**: Address Character Detect bit (bit 8 of received data = 1)
bit 4 **RIDLE**: Receiver Idle bit (read-only)
bit 3 **PERR**: Parity Error Status bit (read-only)
bit 2 **FERR**: Framing Error Status bit (read-only)
bit 1 **OERR**: Receive Buffer Overrun Error Status bit.
bit 0 **URXDA**: Receive Buffer Data Available bit (read-only)
    (Receive buffer has data, at least one more character can be read)

UART is a "dumb" device
Early applications used character data - ASCII
ASCII - only requires 7 bits

Need additional characters
Unicode Project -
Current standard Unicode 6.0
Assigns a code point to every character - U+0000 to U+10FFFF - using 21 bits

ASCII - 0 to 127 (decimal) or 0000 0000 to 0111 1111 (binary)

UTF-8 - a way to encode the characters

1. In UTF-8 ASCII characters are encoded with the 7 least significant bits with msb=0.
2. ALL UCS characters larger than U+007F are encoded as a sequence of two or more bytes.
3. The first byte of a multi-byte sequence indicates how many bytes to follow:

0XXX XXXX - ASCII 7 data bits	0x00 - 0x7F
110X XXXX - 10XX XXXX - 11 data bits	0x80 - 0x07FF
1110 XXXX - 10XX XXXX - 10XX XXXX - 16 data bits	0x800 - 0xFFFF
1111 0XXX -10XX XXXX - 10XX XXXX - 10XX XXXX - 21 data bits	0x10000 - 0x1X XXXX